–Project Πολυδιάστατων Δομών Δεδομένων

Convex Hull

Καραγιάννης Λεωνίδας Ιωάννης 1064036

[st1064036@ceid.webmail.gr](mailto:st1064036@ceid.webmail.gr)

Το project χωρίζεται σε 4 μέρη, το implementation και το visualization σε 2 διαστάσεις και το implementation και visualization σε 3 διαστάσεις.   
Για την εκτέλεση του κώδικα σε 2 ή 3 διαστάσεις υπάρχουν ξεχωριστά αρχεία.

**Αρχείο randomizer.py**

Περίεχει κώδικα για να φτιάχνει και να γεμίζει αρχεία με συντεταγμένες σημείων 2 ή 3 διαστάσεων.

Συναρτήσεις:

* create2Dpoints(start, end, num): Παίρνει ως όρισμα το εύρος των τιμών ως start και end και num το πλήθος των σημείων των οποίων τις διαστάσεις θέλουμε να κάνει generate.
* Create3Dpoints(start, end, num): Ίδια με την create2Dpoints με την μόνη διαφορά το όνομα του αρχείου που φτιάχνει.

**Driver Code:**

1. Ζητάει από τον χρήστη το πλήθος των σημείων, το πλήθος των διαστάσεων των σημείων και το εύρος των τιμών για τις συντεταγμένες
2. Ανάλογα με το πλήθος των διαστάσεων, καλεί την αντίστοιχη συνάρτηση μεταξύ των create2Dpoints και create3Dpoints

**1ο Μέρος – 2 Διαστάσεις**

Για τις 2 διαστάσεις χρησιμοποιήθηκε ο αλγόριθμος [Gift Wrapping ή Jarvis March](https://en.wikipedia.org/wiki/Gift_wrapping_algorithm)

**Αρχείο 2D-final.py**

Περιέχει τον κώδικα για την εύρεση και εικονοποίηση του convex hull σημείων 2 διαστάσεων.

Συναρτήσεις & κλάσεις:

* **Poin**t: Είναι μία κλάση με χαρακτηριστικά τα x και y, τα οποία αποτελούν τις συντεταγμένες των δισδιάστατων σημείων.
* **left\_most\_point(points):** Παίρνει ως είσοδο μία λίστα points τύπου Point και επιστρέφει το index του αριστερότερου σημείου στο πεδίο της λίστας points.
* **orientation(p,q,r):** Παίρνει ως είσοδο 3 σημεία, εφαρμόζει τους τύπους εξωτερικού γινομένου για να βρει την κατεύθυνση της στροφής και επιστρέφει έναν ακέραιο.

0 εάν τα 3 σημεία είναι συνευθειακά

1 έαν η στροφή είναι αυτής της φοράς του ρολογιού

2 εάν η στροφή είναι αντίστροφη της φοράς του ρολογιού

* **convexHull(points,n):** Παίρνει ως είσοδο μία λίστα τύπου Point και το μήκος της λίστας των σημείων.

1. Εξετάζει εάν έχουμε τουλάχιστον 3 σημεία για να μπορεί να οριστεί convex hull σε 2 διαστάσεις
2. Βρίσκει το αριστερότερο σημείο καλώντας την left\_most\_point(points)
3. Δημιουργεί μία λίστα στην οποία θα βάλουμε τα σημεία του convex hull
4. Ξεκινώντας από το αριστερότερο σημείο, κινούμαστε αντίστροφα της φοράς του ρολογιού έως ότου να ξαναφτάσουμε στο αρχικό σημείο
5. Ελέγχει για κάθε σημείο που περνάει εάν η στροφή που έκανε για να το επισκεφθεί ήταν αντίστροφη ή σύμφωνη της φοράς του ρολογιού. Κρατάει το «πιο αντίστροφη της φοράς του ρολογιού» . Εἀν κάποιο σημείο που επισκεφθεί είναι πιο αντίστροφο της φοράς, τους αλλάζει θέση.
6. Τυπώνει όλα τα σημεία
7. Τυπώνει τα σημεία του convex hull
8. Επιστρέφει μία λίστα τύπου Point που περιέχει τα σημεία που αποτελούν το convex hull

* **fileToList():** Δεν έχει όρισμα.

1. Δημιουργεί μία λίστα στην οποία θα βάλουμε τις συντεταγμένες των σημείων που θα διαβάσουμε από αρχείο
2. Ανοίγει το αρχείο στο ίδιο directory με όνομα 2Dpoints.txt και προσθέτει κάθε γραμμή του αρχείου στην λίστα των συντεταγμένων
3. Επιστρέφει την λίστα με τις συντεταγμένες

* **get\_2d\_point\_coordinates(coordinates):** Δέχεται ως όρισμα μία λίστα και βάζει σε 2 λίστες (x\_coordinates, y\_coordinates) τα στοιχεία της λίστας coordinates ανά 2. Επιστρέφει τις λίστες αυτές
* **draw\_2d\_convex\_hull(points, convex\_hull):**

1. Παίρνει ως ορίσματα μία λίστα τύπου Point με όλα τα σημεία και μία λίστα τύπου Point με τα σημεία του Convex Hull
2. Προσθέτει τα σημεία της κάθε μίας λίστας σε λίστες συντεταγμένων για να μπορέσει να τα κάνει scatter σε γραφική παράσταση.
3. Προσθέτει τα αρχικά στοιχεία της λίστας convex\_hull στο τέλος των λιστών με τις συντεταγμένες του convex hull για να «κλείσει ο κύκλος» όταν γίνει η γραφική παράσταση
4. Τυπώνει όλα τα σημεία στο επίπεδο
5. Τυπώνει τα σημεία του convex hull σε κόκκινο
6. Ενώνει τα σημεία του convex hull με κόκκινη γραμμή

* **Driver Code:**

1. Καλεί την fileToList() σε μία λίστα συντεταγμένων coordinates
2. Βάζει τις συντεταγμένες από την coordinates σε 2 λίστες συντεταγμένων x και y
3. Αντιστοιχεί τις συντεταγμένες των λιστών σε συντεταγμένες σημείων, τα οποία προσθέτει στην λίστα points τύπου Point
4. Καλεί την draw\_2d\_convex\_hull με όρισμα την λίστα points και για λίστα σημείων του convex hull καλεί την convexHull με ορίσματα την point και το μήκος της.

**2ο Μέρος – 3 Διαστάσεις**

Για τις 3 διαστάσεις υλοποιήθηκε ο αλγόριθμος [QuickHull](https://en.wikipedia.org/wiki/Quickhull).

Τα βασικά βήματα είναι τα ακόλουθα:

1. Δημιουργούμε το αρχικό τετράεδρο το οποίο θα λειτουργήσει ως βάση. Για να γίνει αυτό αρχικά υπολογίζουμε τα μέγιστα και ελάχιστα σημεία σε όλους τους άξονες. Από αυτά επιλέγονται τα 2 σημεία με την μεγαλύτερη μεταξύ τους απόσταση και σχηματίζουμε μία γραμμή. Μετά βρίσκουμε το σημείο με την μέγιστη απόσταση από την γραμμή και σχηματίζουμε ένα τρίγωνο. Μετά βρίσκουμε το πιο απομακρυσμένο σημείο από το επίπεδο του τριγώνου και φτιάχνουμε ένα τετράεδρο.
2. Μετά διαιρούμε τα σημεία στις 4 πλευρές του τετραέδρου έτσι ώστε τα σημεία να είναι εκτός της κάθε επιπέδου. Αυτό γίνεται παίρνοτας το εσωτερικό γινόμενο της νόρμας με την φορά του ρολογιού του επιπέδου στην Τρίτη γραμή ενώνοντας κάθε κορυφή του επιπέδου με το σημείοπ που ελέγχουμε.
3. Αν η απόσταση είναι θετική, προσθέτουμε το σημείο στην to\_do λίστα της κορυφής και το αφαιρούμε από την αρχική λίστα του προβλήματος (η λίστα με όλα τα σημεία που έχουμε να ελέγξουμε)
4. Αφότου το κάνουμε αυτό και για τις 4 πλευρές του τετραέδρου, αν τα σημεία τα οποία έχουν μείνει στην αρχική λίστα του προβλήματος, τα αγνοούμε. Αυτό θα χρησιμοποιηθεί σε μεταγενέστερα βήματα για να αγνοήσουμε τα εσωτερικά σημεία.
5. Μετά από αυτό πρέπει να συνεχίσουμε το πρόγραμμα μέχρι να υπάρχει πλευρά του πολυτόπου που έχει μηδενική λίστα to\_do.
6. Για κάθε πλευρά βρίσκουμε το πιο απομακρυσμένο σημείο με τα βήματα 2 και 3. Μετά πρέπει να βρούμε τον ορίζοντα αυτού του σημείου, δηλαδή τις κορυφές με τις οποίες θα ενωθεί το σημείο.
7. Τρέχουμε έναν DFS από την πλευρά στης οποίας την λίστα to\_do ήταν το σημείο που ελέγχουμε. Η συνέχιση του DFS εξαρτάται από το εάν το σημείο είναι στην λίστα to\_do της πλευράς. Αυτό θα μας δώσει το τελικό σετ άκρων οι οποίες αποτελούν το σετ του ορίζοντα.
8. Οι άλλες κορυφές της to\_do λίστας χρειάζεται να τους επανα-αναθετηθούν νέες πλευρές του κώνου που δημιουργήθηκε.
9. Συνεχίζουμε μέχρι να μην υπάρχουν άλλα σημεία.

Με την χρήση των κατάλληλων βιβλιοθηκών της Python, το visualization στις 3 διαστάσεις γίνεται με τον χρωματισμό των σημείων του Convex Hull σε διαφορετικό χρώμα από αυτό των υπολοίπων. Καθώς σε μεγάλο αριθμό σημείων είναι δύσκολο να διακριθεί εάν όντως έχουμε πάρει τα σωστά σημεία στο αποτέλεσμα, υπάρχει ένα test file, το οποίο περιλαμβάνει τις συντεταγμένες ενός κύβου και μερικών ακόμα σημείων εντός αυτού. Έτσι, φαίνεται ξεκάθαρα πως τρέχει ορθά ο αλγόριθμος.   
Κατά την εκτέλεση του κώδικα για τις 3 διαστάσεις, εμφανίζεται παράθυρο με δυνατότητα διαδραστικής περιστροφής του πεδίου για να φανούν λεπτομέρειες   
  
  
Όλα τα αρχεία κώδικα περιλαμβάνουν πλήρη σχολιασμό και εξήγηση της λειτουργίας τους.   
Τρέχει σε python 3.7.